

CASE STUDY: Customer Churn with H2O

Customer churn describes the problem that customers may leave a business. The reasons for leaving can be manifold but some of the most common are:

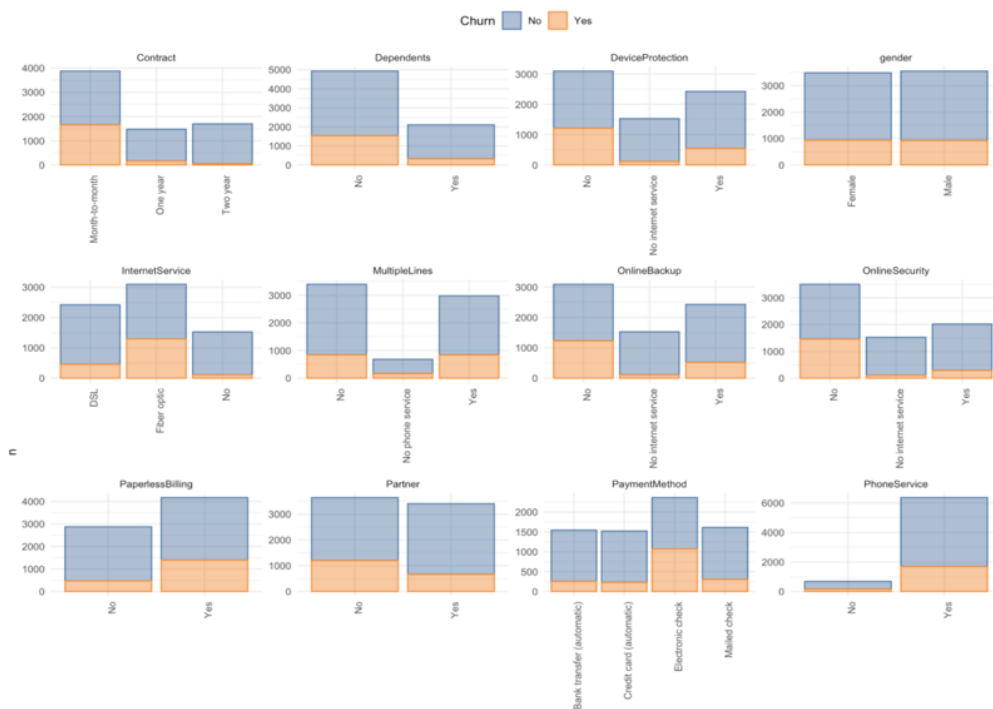
- dissatisfaction with the product, service, etc.
- too expensive
- don't need the service any longer

Customer churn models generally fall into two categories:

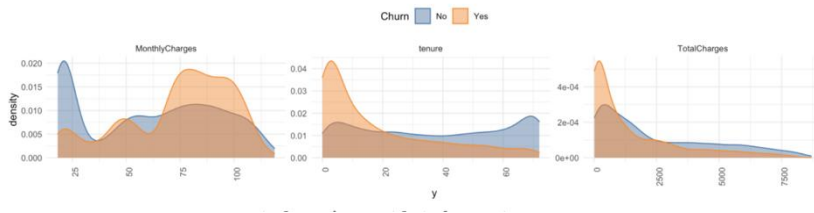
1. Aim to identify customers who are on the brink of leaving, so that appropriate measures can be taken to try and retain them (as a function over time).
2. Aim to identify customer segments who are more likely to churn after a certain period of time (e.g. men being more likely to churn than women, or younger people changing contracts more often than older people, etc.).

Simple customer churn models use the customer's information (demographic information, services that customers have signed up for and account information) to build a target marketing that predicts each customer's likelihood of churning in 3, 6, and 9 months so that marketing campaigns can be focused on them rather than the entire portfolio, which helps directly reaching a certain population, bringing in a certain lead pool, concentrating on a certain market segment, and making financial savings.

More advanced models will also try to classify the reason for potential churn. Depending on the reason, specific actions can be taken to try and prevent the customer from churning (e.g. offering them a better deal when you think they might leave due to finding the service too costly).

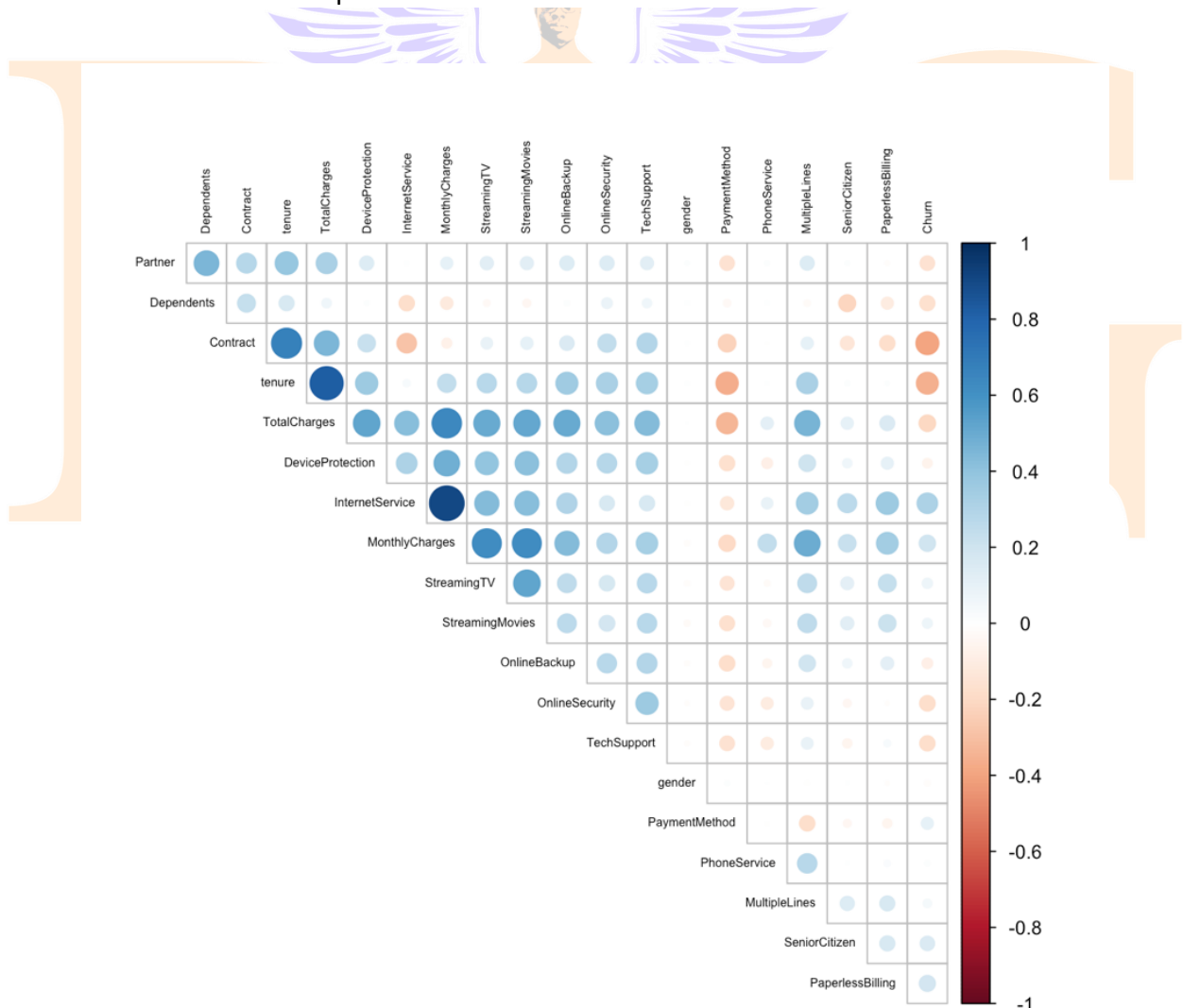


- Plot numerical features (demographic information, services that customers have signed up for and account information):



Before initialize H2O instance and converting data to H2O frame (`library(h2o) h2o.init(nthreads = -1)`), I am building all featuring engineering (dealing with missing data, creating new variables, ...)

Next, I am splitting the data between train and test sets, and partition the test set into validation and test sets. The train dataset alone is used to do the feature selection because performing feature selection with the entire dataset would result in bias in the predictions.



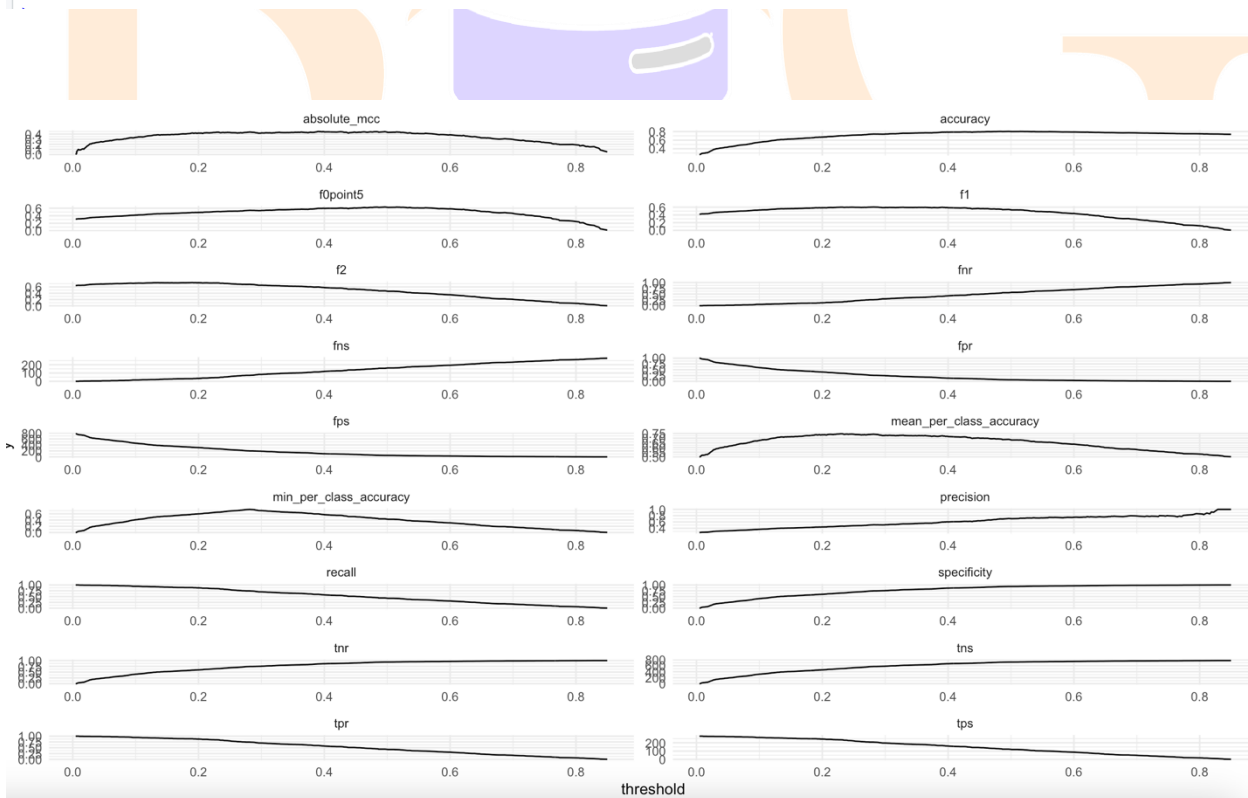
Let convert the final categorical featuring engineering including the 3 numerical data to an H2O frame and start the h2o instance.

Split the data between train and test sets, and partition the test set into validation and test sets. Use the AutoML model to train the training dataset and make predictions using the test dataset. H2O AutoML model works well and has a good accuracy of 0.79, auc of 0.97 and sensitivity level of 0.94.

```

> ##Confusion matrix on validation data
> h2o.confusionMatrix(best_model, valid = TRUE)
Confusion Matrix (vertical: actual; across: predicted) for max f1 @ threshold = 0.296503423577205:
  0  1  Error  Rate
0  583 191 0.246770 =191/774
1   73 207 0.260714 =73/280
Totals 656 398 0.250474 =264/1054
>
> h2o.auc(best_model, train = TRUE)
[1] 0.8730512
> h2o.auc(best_model, valid = TRUE)
[1] 0.8158961
> h2o.auc(best_model, xval = TRUE)
[1] 0.8295642
> #Performance and confusion matrix on test data
> perf <- h2o.performance(best_model, test_hf)
> h2o.confusionMatrix(perf)
Confusion Matrix (vertical: actual; across: predicted) for max f1 @ threshold = 0.28126240070171:
  0  1  Error  Rate
0  575 199 0.257106 =199/774
1   72 208 0.257143 =72/280
Totals 647 407 0.257116 =271/1054
>
> #Plot performance
> plot(perf)
> #More performance metrics extracted
> h2o.logloss(perf)
[1] 0.4405679
> h2o.mse(perf)
[1] 0.1424424
> h2o.auc(perf)
[1] 0.8219938

```



Cost/revenue calculation per year

Let assume that

1. Each customer included in a marketing effort will cost the business \$1000 per year in labor costs plus overhead.
2. In more complicated situations, customers could be further divided into revenue groups to determine how "valuable" they are and how detrimental losing them would be. The average yearly revenue per customer is 2000€.
3. Investing in unnecessary marketing doesn't cause churn on its own (A customer who isn't going to churn isn't reacting negatively to the add campaign, which could happen in more complicated situations).
4. Without a customer churn model, the business would accidentally target half of its customers with advertising efforts.
5. The business would lose about 25% of its customers to churn without a customer churn model.

This would mean that compared to no intervention (having an artificial intelligence model to predict the likelihood of churns before sending a marketing campaign) we would have

Revenue <- \$2,000

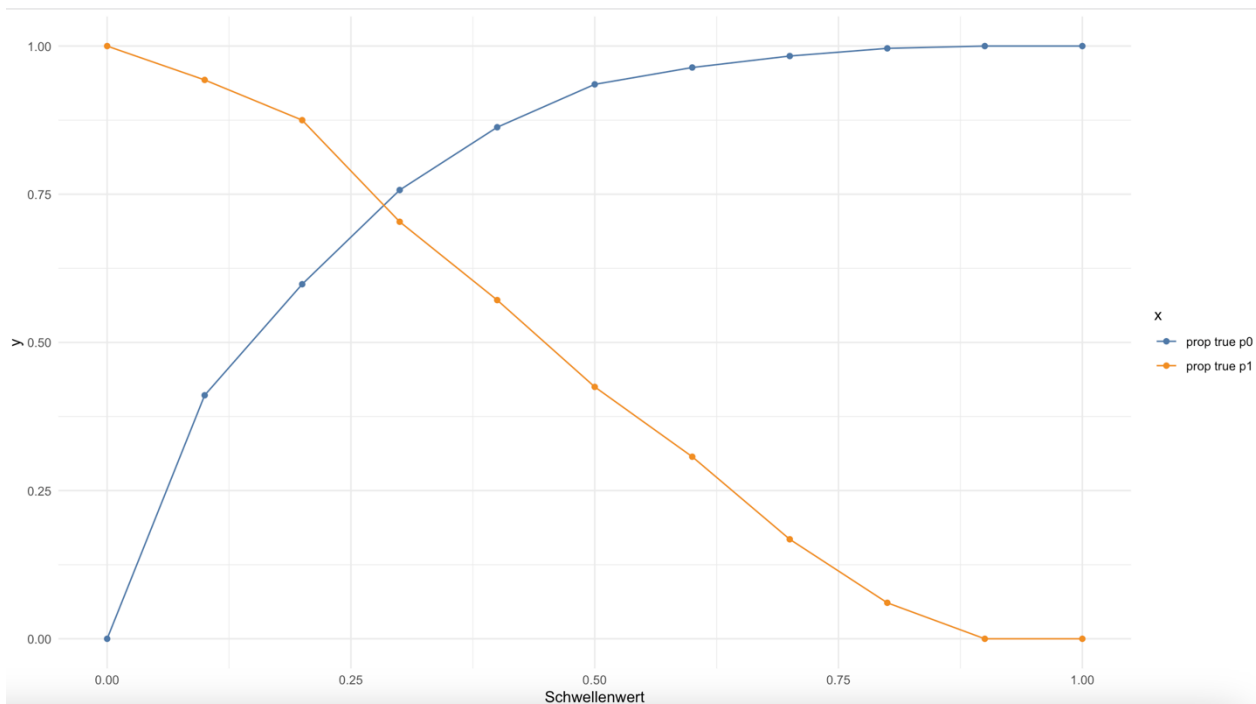
Cost <- \$1,000

customers <- Nbr of customers who churn + Nbr of customers who don't churn.

conversion <- 0.7 customers successful predicted to churn will be targeted by ad campaign.

target_rate <- 0.5 percentage of customers that are randomly targeted for ad campaign

- $\text{prop_p0_correct_pred} == \text{customers who were correctly predicted to not churn did not cost anything: } +/\0
 $\text{prop_p0_correct_pred} == \text{prop_p0_correct_pred} * \text{Nbr of customers that did not churn} * \text{revenue}$
- $\text{prop_p0_wrong_pred} == \text{customers that did not churn but were predicted to churn will be an empty investment: } +/\$0 - \$1,500$
 $\text{prop_p0_wrong_pred} == \text{prop_p0_wrong_pred} * \text{Nbr of customers that did not churn} * (\text{revenue} - \text{cost})$
- $\text{prop_p1_wrong_pred} == \text{customer that were predicted to be successful but churn: } -\2000
 $\text{prop_p1_wrong_pred} == \text{prop_p1_wrong_pred} * \text{Nbr of customers that did churn} * 0$
- $\text{prop_p1_correct_pred} == \text{customers that were correctly predicted to churn:}$
 $\text{pro_p1_correct_pred} == \text{pro_p1_correct_pred} * \text{Nbr of customers that did churn} * ((\text{revenue} * \text{conversion}) - \text{cost})$
 - let's say 100% of those could be submitted correctly by investing into artificial intelligence model: $+\$2,000 - \$1,500$
 - let's say 50% could be submitted correctly by investing into artificial intelligence model: $+\$2,000 * 0.5 - \$1,500$



- Let's play around with some values:

Net win per year (revenue from customers that did not churn-ad costs) == (Nbr of customers that did not churn*revenue) - (customers*cost*target_rate) = \$1,021,000

Net win = prop_p0_correct_pred_x + prop_p0_wrong_pred_x + prop_p1_wrong_pred_x + pro_p1_correct_pred_x

Net win table = net win-net win per year

threshold	net_win	net_win_compared
0.6	1554400	533400
0.7	1553800	532800
0.8	1551800	530800
0.9	1548000	527000
1.0	1548000	527000
0.5	1545600	524600
0.4	1506000	485000
0.3	1438800	417800
0.2	1335000	314000
0.1	1197600	176600
0.0	886000	-135000



Plot Zoom

